Session 605

Moving to WebObjects 5

Melissa Turner, Francois Jouaux, RD Willhoite
Random WebObjects Engineers

Introduction

This session

- Assumes some knowledge of WebObjects
- Discusses conversion issues for those with existing WebObjects 4.5 or 4.5.1 applications

What Is Covered

What you will learn

- Benefits of moving to pure Java in WebObjects 5
- How to migrate to WebObjects 5 with the Java Converter tool
- Improvements and new features in WebObjects 5 Frameworks

What Is Not Covered

What you won't learn

- In depth discussion of WebObjects technologies
  - EOF, Java Client, Direct To Web, XML
- Any technology not mentioned here is covered in other sessions in greater detail

What Is WebObjects 5?

- Essentially WebObjects 4.5 ported to Java
- Full featured and API compliant with WebObjects 4.5 (with a few exceptions)
- Uses WebObjects 4.5 developer tools on Win2K
- New developer tools on Mac OS X
- WebObjects 5 monitoring tools capable of managing WebObjects 4.5.1 instances
- New features

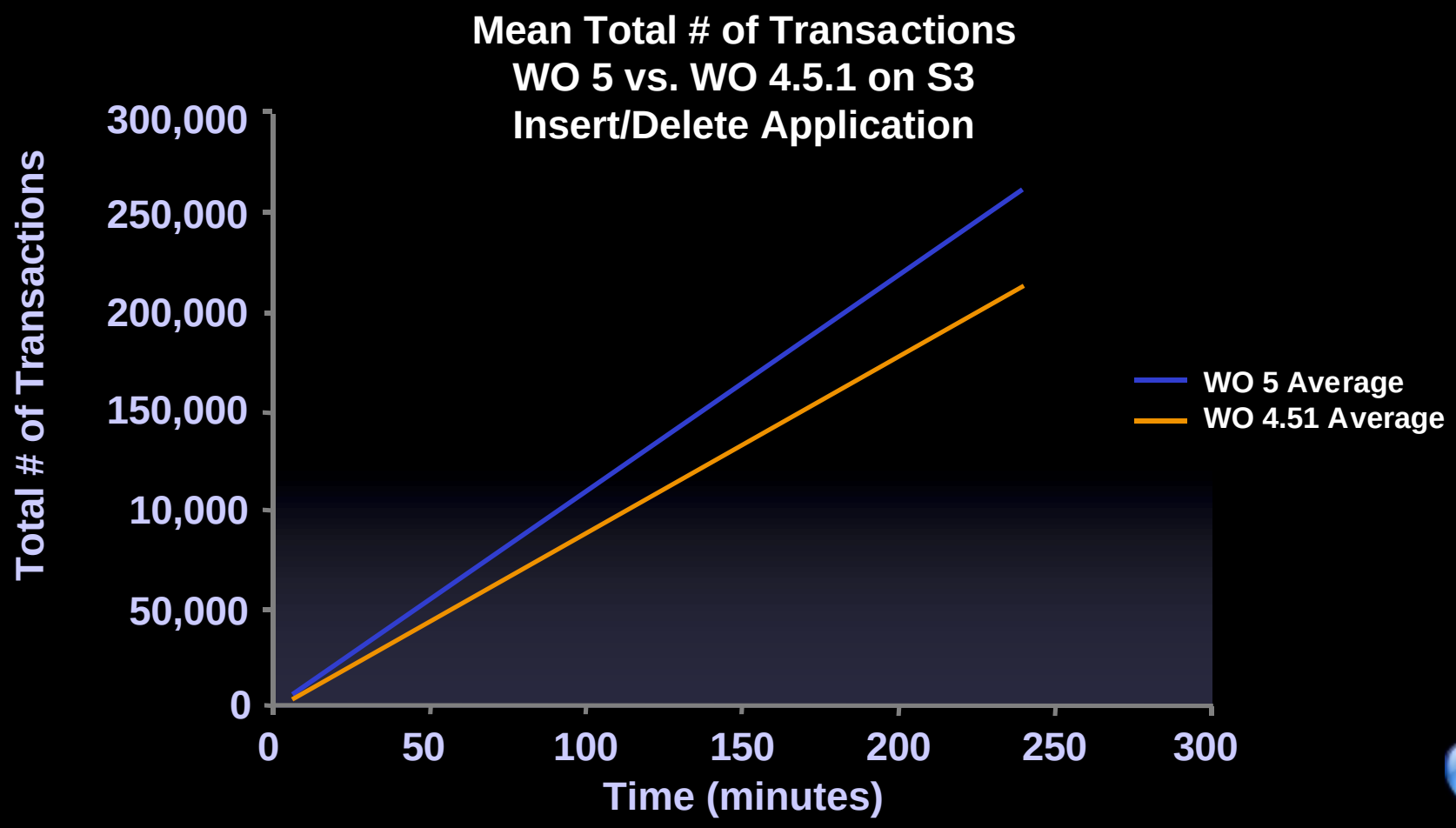
Benefits I

- WebObjects 5 uses JDK 1.3
  - JDK 1.1.8 is included on Windows only for use by the tools, not the runtime
- Allows integration with third-party development tools, debuggers, and optimizers—impossible with bridged Java
- Integrates with any third party-products with a Java interface
- Extended database connectivity through JDBC

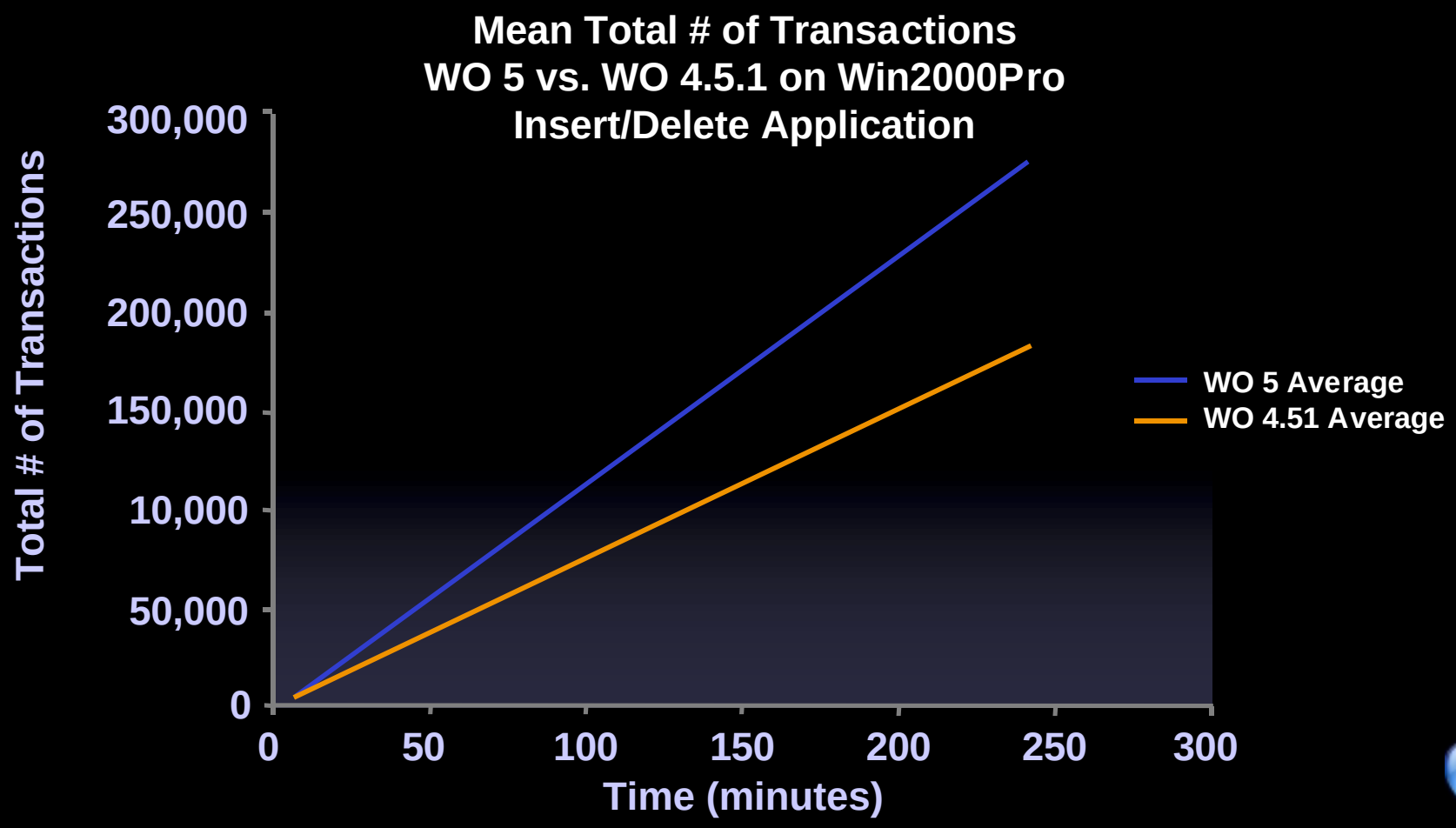
Benefits II

- Development on Mac OS X or Windows 2000
- Deployment on above platforms, plus Solaris
- Ease of portability between deployment platforms
- Performance improvements!

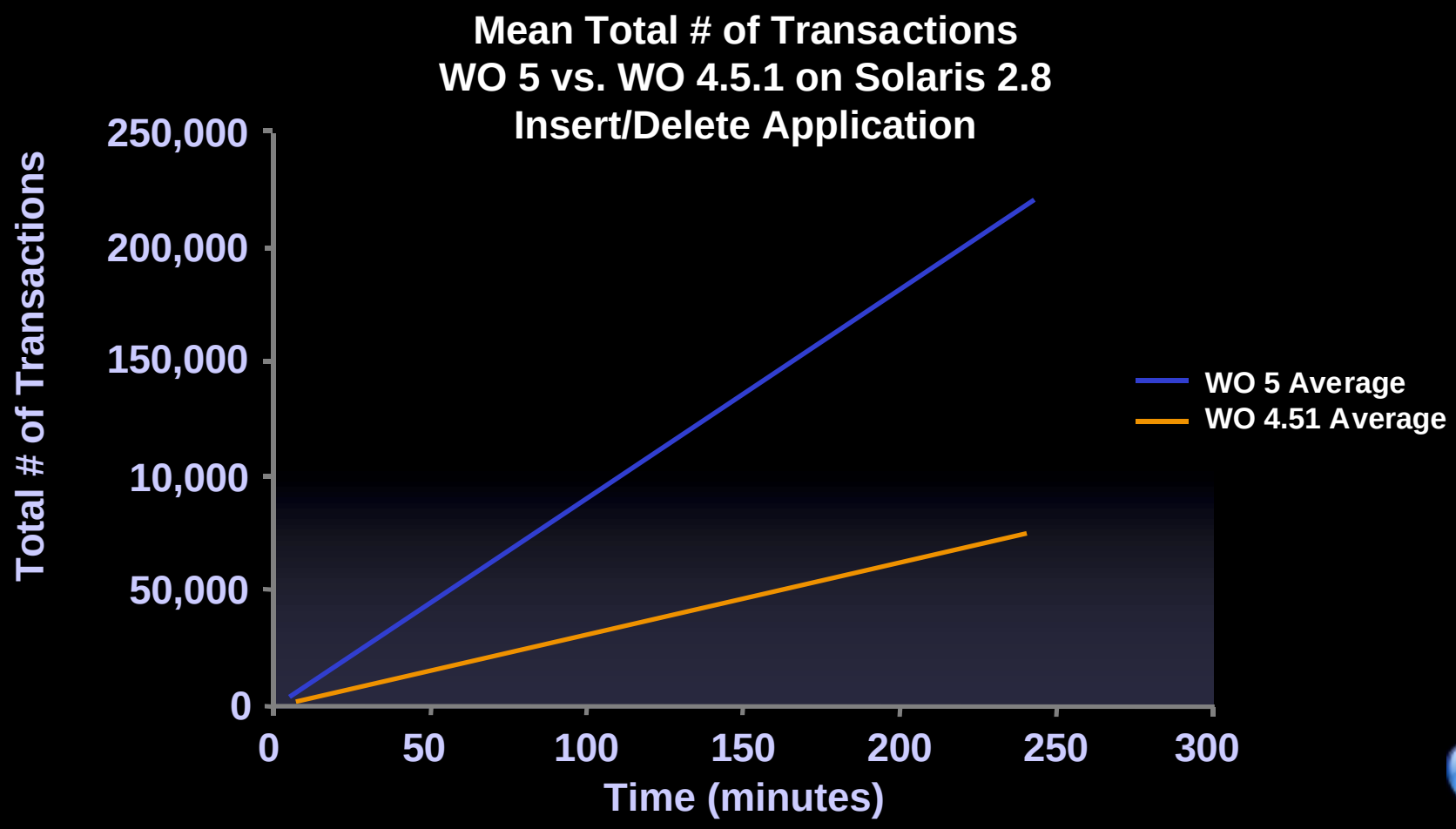
Performance Improvements—Mac OS X



Performance Improvements—Windows



Performance Improvements—Solaris



Q: Who Should Convert?

A: Anyone who…

- Has Objective-C or WebScript projects not yet in production
- Has existing bridged Java projects
- Requires access to more Java standards

Q: Who Should Probably Not Convert?

A: Anyone who…

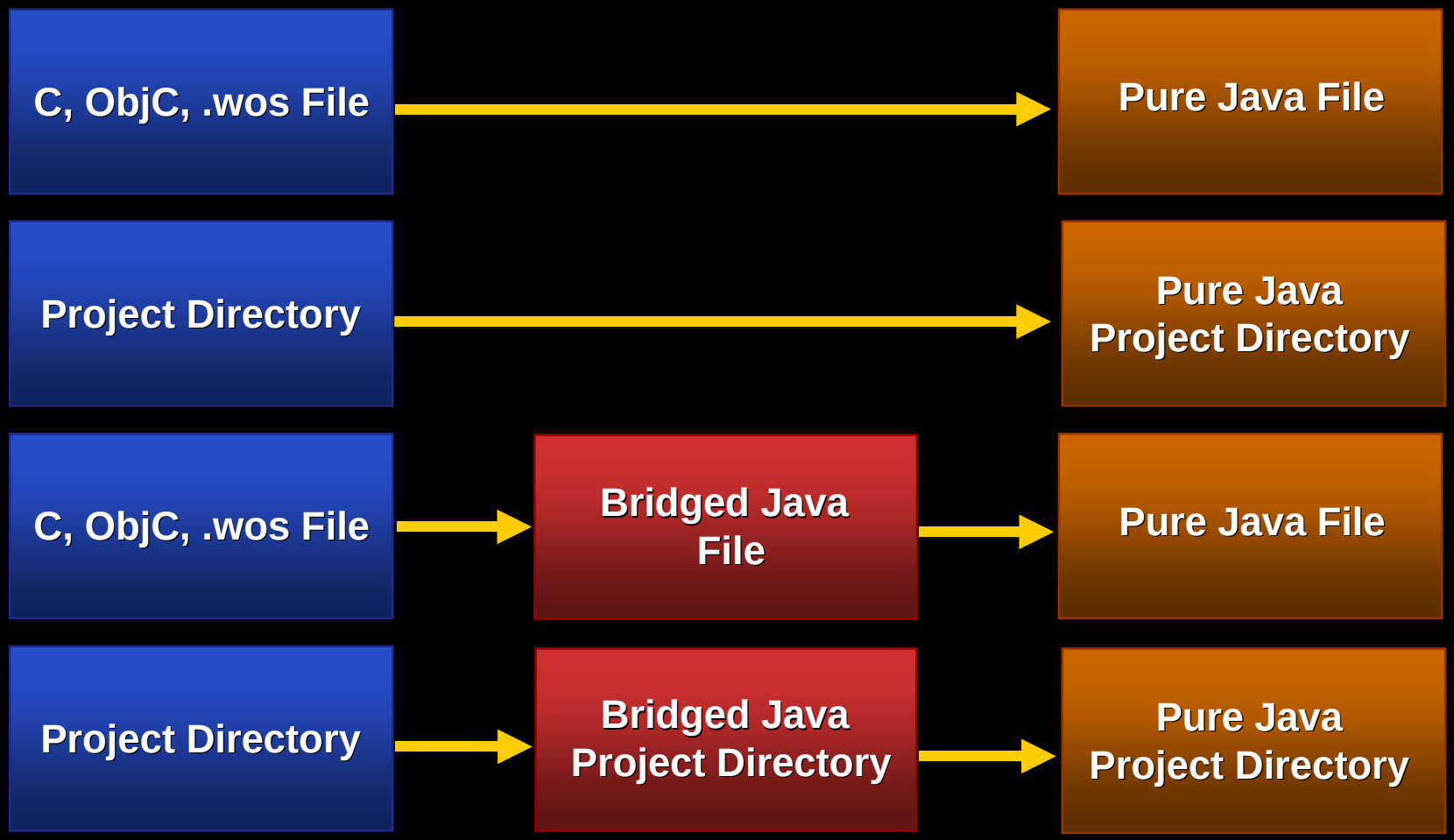
- Has Objective-C projects using lots of categories, posers, or low-level Objective C runtime methods
- Has lots of basic C code
  - Calls to C libraries
  - Pointer manipulation
- Has an app already in production that is not being updated

Q: How Do I Convert?

A: Use the Converter tool

- Located at
  - Win2k: C:/Apple/Library/WebObjects/JavaApplications/JavaConverter.woa
  - Mac OS X: /System/Library/WebObjects/JavaApplications/JavaConverter.woa
- Run from inside .woa directory
  - Win2k:
    ./JavaConverter.cmd [Full path of project directory or file(s)]
  - Mac OS X:
    ./JavaConverter [Full path of project directory or file(s)]

Migrating Using JavaConverter



Advantages of JavaConverter

- Original source code is left untouched
- Conversion problems clearly described in comments tagged **JC_ERROR**, **JC_WARNING** or **JC_INFO**
- Tested against major Objective-C frameworks (WebObjects, EOControl, EOAccess)
- Extensible with customized scripts
- Faster than a manual conversion

Limitations of JavaConverter

- EOModel files are not converted
- APIs in .wod and other archive files are not converted
- Objective-C code that relies on posers, categories, or low level Objective-C runtime is not fully converted
- C code with pointer manipulation and/or using C libraries does not convert well
- The conversion process takes you 80% there!

Demo

Java Converter

Francois Jouaux
Manager, WebObjects Deployment

Conversion

- Convenience classes to facilitate porting to pure Java
  - NSComparator
  - NSBundle
  - WOTimer
- NSCoder/NSCoding are abstract
- NSRunLoop removed
- Changes in user defaults
- Changes in validation mechanism

Conversion: NSComparator

- Abstract class to encapsulate **NSComparisonResult** from Objective C
- Used with NSArray method **sortedArrayUsingComparator()** to sort the elements of the receiving array into a new array
- Or NSMutableArray method **sortUsingComparator()** to sort the elements of an array in place
- When writing new Java code, try to use java.lang.Comparable instead of NSComparator

Conversion: NSBundle

- NSBundle
  - Corresponds to a directory where related resources are stored
  - Maintains some deprecated API to facilitate conversion from Objective-C
  - Functionality extended to handle Properties file, and new APIs for resource finding and loading
- The Java class loader does not support dynamic class loading, so there is no concept of “loadable bundles” in WebObjects 5

Conversion: Renaming and Deletion

- NSTimer → WOTimer
  - No longer used for session timeouts
- NSRunLoop no longer exists
  - Now a private WORunLoop with reduced functionality to handle WOApplication specific tasks, session timeouts, and timers
- No concrete subclass of NSCoder provided
  - Use Java serialization instead

Conversion: User Defaults

- NSUserDefaults is no more
  - Replaced by NSProperties
- These defaults are now deprecated
  - **WODebuggingEnabled**
  - **NSProjectSearchPath**
- These defaults have been renamed
  - **WOCGIAdaptorURL** (WO4.5) → **WOAdaptorURL** (WO5)
  - **WOShouldSendLifebeat** (WO4.5) → **WOLifebeatEnabled** (WO5)

Conversion: Changes in Validation Mechanism

- Validation methods in WebObjects 4.5
  - Take a pointer to the target as their parameter
  - Return an exception object if the target object is invalid
  - If coercion is necessary, the validation method would have a side effect on its parameter

Conversion: Changes in Validation Mechanism II

- Validation methods in WebObjects 5
  - Take an object as the parameter
  - Return the desired, valid value
  - If parameter is invalid, an NSValidation.ValidationException must be thrown
  - Returning null is interpreted as value coercion to the null Object reference
- The Java Converter does not handle this automatically

Conversion: Validation Examples

```
// Objective-C code
- (NSException *) validateStreetAddress: (NSString *)
  address;
// 4.5 Java
public void validateStreetAddress(String address);
// Proper conversion in 5
/* Returns the valid string object or throws
  NSValidation.Exception if the street address string is
  invalid. */
public String validateStreetAddress(String address)
  throws NSValidation.ValidationException;
```

Conversion: Validation Examples

```
// Objective-C code
- (NSException *) validateStreetAddress: (NSString *)
   address;

// 4.5 Java
public void validateStreetAddress(String address);

// Proper conversion in 5

/* Returns the valid string object or throws
   NSValidation.Exception if the street address string is
   invalid. */
public String validateStreetAddress(String address)
   throws NSValidation.ValidationException;
```

Conversion: Validation Examples

```
// Objective-C code
- (NSException *) validateStreetAddress: (NSString *)
   address;

// 4.5 Java
public void validateStreetAddress(String address);

// Proper conversion in 5

/* Returns the valid string object or throws
   NSValidation.Exception if the street address string is
   invalid. */
public String validateStreetAddress(String address)
   throws NSValidation.ValidationException;
```

New Features

- New to WebObjects Foundation Framework
  - NSTimestamp, NSTimestampFormatter, NSTimeZone
  - NSSocketUtilities
  - NSLog
  - NSProperties
  - NSLocking
  - Key Value Coding
  - Serialization

New Features II

- Other WebObjects Java packages
  - WOMailDelivery
  - WOSMIL
  - JavaPlot

Why NSTimestamp?

A: Wanted to ensure functionality of NSGregorianDate in Java

- Ran into some restrictions in JDK
  - java.util.TimeZone lacks historical information
  - java.util.Date and java.util.Calendar make assumptions that made integrating NSTimeZone difficult
- Why not NSCalendarDate?
  - Shows descent from java.sql.Timestamp

NSTimestamp

java.util.Calendar

java.util.TimeZone

NSTimestamp

- Subclass of java.sql.Timestamp
- Replaces all uses of NSCalendarDate
- Provides most of NSCalendarDate functionality that has no equivalent in Java
- Database DATE types now mapped to NSTimestamp
- New reference date: Jan. 1, 1970 UTC instead of Jan. 1, 2001 UTC

NSTimestampFormatter

- Subclass of java.text.Format
- Used to format NSTimestamps into their textual representations and convert textual representations of dates and times into NSTimestamps

NSTimeZone

- Used with NSTimestamp to adjust dates for the proper historical and geographical context
- Subclass of java.util.TimeZone
- Built using the standard zoneinfo files from **ftp://elsie.nci.nih.gov**
- To be used only with NSTimestamp and NSTimestampFormatter (Don't mix with standard Java Date/Time classes)

NSSocketUtilities

- Lets you specify a connection timeout
- If socket creation times out an exception is thrown
- Without these utilities, failure to create sockets can block indefinitely depending on the operating system

NSSocketUtilities: Code Sample

```
// The standard Java way
Socket s = new Socket (address, port, localAddress,
   localPort);

// The NSSocketUtilities way
Socket s;
try {   // wait 1 second for socket creation/connection
   s = NSSocketUtilities.getSocketWithTimeout(address,
            port, localAddress, localPort, 1000);
} catch (IOException) {
   // deal with lack of socket
}
```

NSSocketUtilities: Code Sample

```
// The standard Java way
Socket s = new Socket (address, port, localAddress,
   localPort);


// The NSSocketUtilities way
Socket s;
try {   // wait 1 second for socket creation/connection
   s = NSSocketUtilities.getSocketWithTimeout(address,
            port, localAddress, localPort, 1000);
} catch (IOException) {
   // deal with lack of socket
}
```

NSSocketUtilities: Code Sample

```
// The standard Java way
Socket s = new Socket (address, port, localAddress,
   localPort);


// The NSSocketUtilities way
Socket s;
try {   // wait 1 second for socket creation/connection
   s = NSSocketUtilities.getSocketWithTimeout(address,
            port, localAddress, localPort, 1000);
} catch (IOException) {
   // deal with lack of socket
}
```

NSProperties

- Replaces NSUserDefaults
- Uses standard Java properties mechanism (java.util.Properties)
- Convenience class for merging application properties with standard system properties
- For conversion convenience, some deprecated methods are still available

Properties

- Application properties can be read from five sources (in this order)
  - The system Properties (come with the VM)
  - The user's properties file in ~/.hotjava/Properties
  - The Properties file of any frameworks your application includes,
  - The application's Properties file
  - The command line
- In cases of property contention, the last property read will be used

Properties

- Properties files must be:
  - Named “Properties”
  - In the Resources directory of the built application or framework
  - In the format specified by java.io.Properties
- To specify a property on the command line:

```
/FullPath/HelloWorld.woa/HelloWorld
                    -DWOSessionTimeOut=NO
```

Debugging with NSLog

- Three new classes
  - NSLog
    - Static class to access the WebObjects logging system
  - Logger
    - Abstract class that specifies the core functionality of NSLog
    - Can be subclassed for customized logging
  - PrintStreamLogger
    - Concrete subclass of Logger to log output to a java.io.PrintStream contained by the logger

Debugging with NSLog

- Why use NSLog?
  - Can control scope and granularity of debugging output with debug groups and debug levels
  - Can redirect output as desired
    - To a java.io.PrintStream
    - Email
  - Can print the thread that did the logging and the timestamp
    - Useful for debugging synchronization problems
  - System.out.println() can lead to freezes in deployed applications which are launched using exec()

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
// Create a custom NSLog group
public static final long MyDebugGroup = 1L<<40;

// Using NSLog from the command line
// Parameter should be passed as -DMyDebugging=true
boolean shouldDebug = Boolean.getBoolean("MyDebugging");
If (shouldDebug) {
   NSLog.debug.setIsVerbose(true);
   NSLog.out.setIsVerbose(true);
   NSLog.err.setIsVerbose(true);
   NSLog.allowDebugLoggingForGroups(MyDebugGroup);
   NSLog.setAllowedDebugLevel(NSLog.debugLevelInformational);
}
```

NSLog: Code Samples

```
//Redirecting output to a path passed in as
//  -DMyOutputFile=/tmp/out.txt
String outputPath = System.getProperty("MyOutputFile");
if (outputPath != null) {
   java.io.PrintStream aStream =
            NSLog.printStreamForPath(outputPath);
   if (aStream != null) {
            ((NSLog.PrintStreamLogger)
                     NSLog.debug).setPrintStream(aStream);
            // Also set the NSLog.out and NSLog.err
   }
}
```

NSLog: Code Samples

```
//Redirecting output to a path passed in as
//  -DMyOutputFile=/tmp/out.txt
String outputPath = System.getProperty("MyOutputFile");
if (outputPath != null) {
    java.io.PrintStream aStream =
            NSLog.printStreamForPath(outputPath);
    if (aStream != null) {
            ((NSLog.PrintStreamLogger)
                    NSLog.debug).setPrintStream(aStream);
            // Also set the NSLog.out and NSLog.err
    }
}
```

NSLog: Code Samples

```
//Redirecting output to a path passed in as
//  -DMyOutputFile=/tmp/out.txt
String outputPath = System.getProperty("MyOutputFile");
if (outputPath != null) {
    java.io.PrintStream aStream =
            NSLog.printStreamForPath(outputPath);
    if (aStream != null) {
            ((NSLog.PrintStreamLogger)
                    NSLog.debug).setPrintStream(aStream);
            // Also set the NSLog.out and NSLog.err
    }
}
```

NSLog: Code Samples

```
//Redirecting output to a path passed in as
//  -DMyOutputFile=/tmp/out.txt
String outputPath = System.getProperty("MyOutputFile");
if (outputPath != null) {
    java.io.PrintStream aStream =
            NSLog.printStreamForPath(outputPath);
    if (aStream != null) {
            ((NSLog.PrintStreamLogger)
                    NSLog.debug).setPrintStream(aStream);
            // Also set the NSLog.out and NSLog.err
    }
}
```

NSLog: Code Samples

```
//Redirecting output to a path passed in as
//  -DMyOutputFile=/tmp/out.txt
String outputPath = System.getProperty("MyOutputFile");
if (outputPath != null) {
   java.io.PrintStream aStream =
            NSLog.printStreamForPath(outputPath);
   if (aStream != null) {
            ((NSLog.PrintStreamLogger)
                     NSLog.debug).setPrintStream(aStream);
            // Also set the NSLog.out and NSLog.err
   }
}
```

Demo

NSLog

RD Willhoite
WebObjects Engineer

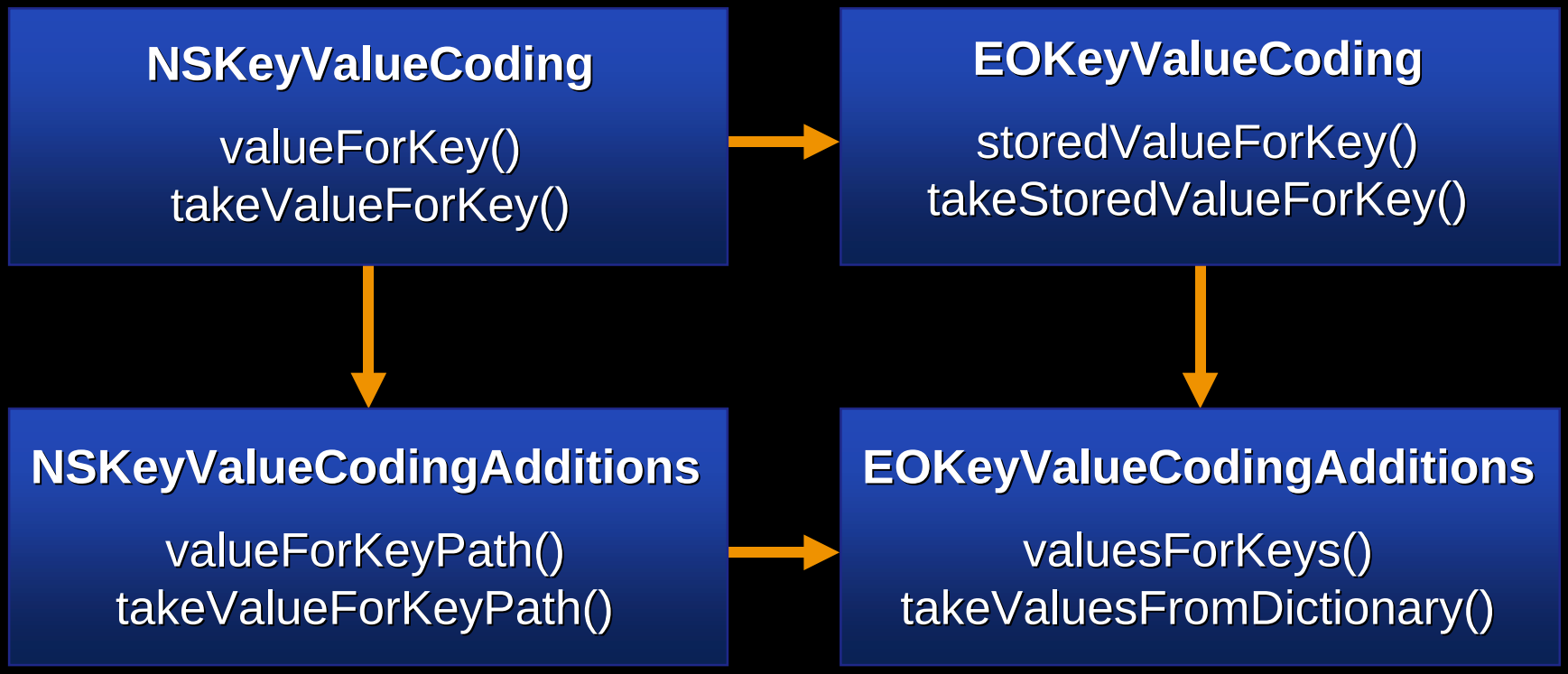
NSLock

- 3 Classes: NSLock, NSMultiReaderLock, NSRecursiveLock
- Conform to the NSLocking interface
- Facilitate conversion from previous WebObjects versions
- Can coordinate locking objects over a longer period of time, independent of the code execution path
- Allow developers to do things that are difficult to achieve using only synchronize

NSKeyValueCoding

- A unified way to access properties of different types
  - Instance variables
  - Methods
- This is the old EOKeyValueCoding split up in multiple parts
  - 4 abstract interfaces defining methods to be implemented
  - Error handling interface
  - Default implementations

NSKeyValueCoding



NSKeyValueCoding

- KeyValueCoding.DefaultImplementation
  - A convenient way to get the default behavior if you want to add KeyValueCoding interface to a class
- KeyValueCoding.Utility
  - A convenient way to get the default behavior for a class that does not conform to this KVC interface

Serialization

- NSCoder and NSCoding now replaced with native Java serialization scheme
- All classes that implemented NSCoding are now Serializable
- See PersistentSessionStore as an example of the new Java serialization scheme

Serialization: Advantages

- Java developers do not need to understand another archiving scheme
- The Java runtime framework does all the work now

WOMailDelivery

- Now based on sun.net.smtp.SmtpClient
- This protocol does not support cc: or bcc:, so cc'ed people also appear in the "to:" field
- Now allows mailed pages to submit forms and see images because absolute URLs are generated

JavaPlot

- Now AWT based
- All AWT fonts are available
- Images are anti-aliased
- Bug fixes and improvements
- Subject to limitations in the AWT

WOSMIL Framework

- Now in the official release of WebObjects 5
- Allows WebObjects applications to deliver SMIL content
- Implements SMIL 1.0 spec from W3C
- Allows you to generate dynamic multimedia content, playable with QuickTime player
- Integrated with WebObjects Builder via palettes for a drag and drop experience

Summary

- Mostly the same as WebObjects 4.5
- We provide JavaConverter to help you migrate
- New features to make life easier

WebObjects Lab

- Located downstairs in Room K
- Lab hours
  - Monday 12:00pm–6:00pm
  - Tuesday 10:30am–6:00pm
  - Wednesday 10:30am–6:00pm
  - Thursday 10:30am–6:00pm
  - Friday 10:30am–3:30pm

☑ Vote for WebObjects as Best App Server
http://developer.apple.com/webobjects

Roadmap

606 Direct to Java Client	Room A1 **Wed., 3:30pm**
607 Practical Uses of XML with WebObjects	Room A1 **Wed., 5:00pm**
611 Deploying WebObjects Applications	Room A1 **Thurs., 10:30am**
FF111 Feedback Forum: WebObjects	Room A1 **Fri., 3:30pm**

Who to Contact

Toni Trujillo Vian
Director, WebObjects Engineering
webobjects@apple.com

Bob Fraser
WebObjects Product Manager
webobjects@apple.com

iServices Consulting, Integration, Training & Certification
(800) 848-6398
iservices@apple.com

For more information:
www.apple.com/webobjects
developer.apple.com/webobjects
www.apple.com/iservices
www.info.apple.com/webobjects

Subscribe to:
webobjects-announce@apple.com

Q&A

Toni Trujillo Vian
Director, WebObjects Engineering
webobjects@apple.com

http://developer.apple.com/events/urls.html

WWDC

Worldwide Developers Conference 2001